

Unix User Settings

Makefile

Core Dump

GDB

CIT 593

CIT593

1/13

Shell Variables

- Predefined Variables examples
 - HOME
 - Contains user home directory path
 - PATH
 - Contains all the search paths for your shell
 - SHELL
 - Tells which is your default shell
- In order use these variables you need to put **\$** sign in front of the variable name. E.g.
 - echo \$HOME -> prints your home directory path (even if you are not in your home directory)
 - **echo** is shell command to print
 - If you do not put the \$ sign then it will just print "HOME"

CIT593

2/13

.cshrc or .bashrc file

- Are files for customizing shells
 - .cshsrc for csh or tcsh shell
 - .bashrc for bash shell
- These files contains commands, variable definitions and aliases used any time the type shell is run.
 - When one logs in, for e.g. C shell starts by reading the *.cshrc* file, and sets up any variables and aliases

CIT593

3/13

Customizing .cshrc Example

```
# if you want to add directories to your search path, add them before
# the "." in the line below. For example,
#
# set path = ( $path ~ ~/bin ~/scripts . )

set path = ($path /usr/java/jdk1.5.0/bin .)

# to be automatically logged out after 30min of inactivity:
# set autologout=(30)
unset autologout

#####DIANA#####
if ($?prompt) set prompt = "palsetia%c04:"
alias c clear
alias p pine
```

CIT593

4/13

After editing .cshrc

- At the prompt type: source .cshrc
 - This will allow the changes to take place effectively
 - Otherwise, the changes will take place next time you login
 - Because .cshrc script is executed once i.e. executed when user logs in

CIT593

5/13

Core Dump

- A core dump is the name often given to the **recorded state of the working memory of a computer program** at a specific time
 - generally when the program that has terminated abnormally
- Is a simply a **binary** file with the sequence of bytes or words containing the memory image of a particular process.
- Name of file is “**core**” without any extensions and the file appears in the current working directory

CIT593

6/13

Core Dump (contd..)

Abnormal Program Termination

- Often buffer overflows, where a programmer allocates too little memory for incoming or computed data
- Access to null pointers, a common coding error when an unassigned memory reference variable is accessed

Uses of core dump

- A runtime error such as “Segmentation Fault” does not describe as to where does the problem lie
- Core dump can help reveal where the fault is occurring
 - Programmer can try to interpret the file but needs enough knowledge of structure of the programs memory use
 - Special Programs such as GDB debugger or dump analyzers can relieve the programmer from reading core file which is in hex bytes

CIT593

7/13

Core Dump (contd..)

Issue with core file

- Core file is often very large and takes up a lot of disk space
- One can avoid exceeding the disk quota with a large core file by limiting the core dump size

csh/tcsh shell

- limit coredumpsize 0
- limit coredumpsize 10240 # Limit core dumps to 10 megs
- unlimited coredumpsize # Allow unlimited-sized coredumps

CIT593

8/13

make

make is a compilation tool

- make compiles, assembles, and links your source files
- Think of it as the lcs file in PennSim simulator
- The file you write is called "makefile" without any file extensions
 - **makefile** is read by program **make**
- Then at prompt type : make *targetname*

CIT593

9/13

Example: makefile

```
#is a comment
COMPILER=gcc
LIBS=-lm #needed to compile Math.h
OPTIONS=-Wall
NAME1=hypotenuse
NAME2=main
NAME3= hypo

all:
    $(COMPILER) $(LIBS) $(OPTIONS) $(NAME1).c $(NAME2).c -o
    $(NAME3)

# To remove executables or .o files generated
# Type: make clean
clean:
    rm -f *.o $(NAME3)
CIT593
```

10/13

makefile: Rule Syntax

A rule tells make two things:

- when the targets are out of date
- how to update them when necessary.

In general, a rule looks like this:

```
targets : prerequisites
    <"t">command
```

... or like this:

```
targets : prerequisites ; command command ...
```

CIT593

11/13

makefile: Rule Syntax (contd..)

- The *targets* are named sections (like a LABEL) to tell make what to execute
 - There can be more than one target
 - When use make, specify the target you want
 - If *targetname is not provided*, then the make executes the first target in the file
 - The *command* lines start with a tab character ('\t')
 - The first command may appear on the line after the prerequisites, with a tab character, or may appear on the same line, with a semicolon
 - You may split a long line by inserting a backslash followed by a newline
 - But this is not required, as make places no limit on the length of a line in a makefile

CIT593

12/13

GDB Debugger

GDB stands for **GNU Debugger**

- GDB is a Unix terminal debugger, which can execute your program in a sandbox and allow you to analyze it during runtime
- Most helpful to catch segmentation fault error
 - Can pin-point exactly where in your code it happening
- Follow the GDB tutorial on how to use GDB tool